Universidad Zaragoza

1542

Trabajo Fin de Grado

Soluciones basadas en redes neuronales para el problema de muchos cuerpos cuánticos

Autor

Mikel Escobar De Carlos

Directores

David Zueco Lainez

David González Rojas

FACULTAD DE CIENCIAS

2020

Índice

1. Introducción

Durante los últimos años las técnicas de machine learning han atraído la atención de las grandes empresas debido a su versatilidad y capacidad de transformar sus productos y unidades de negocio. Esto ha dado una importante publicidad a este tipo de tecnologías que unen las matemáticas y las ciencias de la computación provocando su popularización en diversas áreas. Podemos encontrar modelos de machine learning en los sistemas que nos recomiendan vídeos o películas en los momentos de ocio o en la cámara de fotos de nuestro teléfono móvil para detectar el tipo de escena y establecer automáticamente los ajustes óptimos a la hora de fotografiar, o para poder desbloquear el dispositivo con nuestra cara.

Más allá de las aplicaciones que todos conocemos en nuestro día a día, este tipo de técnicas se han difundido ampliamente dentro del ámbito científico, donde fueron en primer lugar diseñadas para tareas como clasificación de galaxias [1] o clusterización de datos. Esto está permitiendo por ejemplo al personal sanitario procesar imágenes de forma automática detectando tumores u otras enfermedades en radiografías con una precisión que supera a la de un médico humano [5]. Otros campos de la física que en un principio no se vieron involucrados con estas técnicas han explorado recientemente su aplicabilidad y muchos investigadores han visto mejorado o acelerado su trabajo apoyándose en el machine learning. Ejemplos de ello son los científicos trabajando en el LHC [4] o en temas de materia condensada. Este tipo de técnicas les permite gestionar y procesar de forma muy eficiente los inmensos volúmenes de datos que se generan diariamente en los diversos centros de investigación del mundo. Esta revolución viene de la mano del enorme impulso al desarrollo de estas tecnologías por parte las grandes compañías tecnológicas del planeta, facilitando el acceso a información y librerías que acercan el machine learning a un público mucho más amplio. Por otra parte, debemos de tener en cuenta la gran evolución del hardware y las estructuras de computación en la nube que ha tenido lugar en la última década. Este aumento en la potencia computacional ha habilitado la posibilidad de construir modelos de una complejidad antes impensable y generalizar aplicaciones como la visión por computador o el procesamiento del lenguaje natural.

En este trabajo vamos a tratar de emplear técnicas de machine learning basado en redes neuronales para resolver de forma eficiente sistemas de muchos cuerpos cuánticos. La resolución numérica de sistemas de muchos cuerpos (cuánticos o clásicos) implica una complejidad de cálculo del orden de al menos $\mathrm{O}(N^2)$. Para afrontar esta problemática se emplean métodos aproximados como la aproximación de campo medio. Cuando entramos en sistemas de cuerpos cuánticos los posibles estados del sistema crecen de forma exponencial con el número de elementos, provocando que el coste computacional para resolver de forma exacta este tipo de sistemas sea prohibitivo a partir de unas pocas decenas de cuerpos.

Un ejemplo de ello son los resultados presentados por Google en [7] donde, a finales de 2019 y con todos los recursos que dispone una compañía de este tamaño, eran sólo capaces de almacenar en memoria un estado de hasta 43 qubits para simular su evolución de forma exacta.

En definitiva, el problema general al que ha de enfrentarse cualquier persona tratando de resolver sistemas de quantum many-body es la inabarcable dimensionalidad del espacio de Hilbert, incluso para sistemas con números de partículas mucho menores que los sistemas experimentales. Esta complejidad inherente a la función de ondas de un sistema de muchos cuerpos se ha tratado mediante un amplio abanico de métodos clásicos tanto perturbativos como no perturbativos, basados en la integral de camino o en procesos de Monte Carlo y han establecido la base de métodos variacionales para la representación de esta función many body. Esta problemática es la que tratamos de afrontar también desde una nueva representación variacional basada en redes neuronales que ya ha demostrado su viabilidad en publicaciones recientes. Siendo la base de la inspiración para comenzar este trabajo las publicaciones [3] y [2].

2. Modelo de Bose-Hubbard

Con el fin de aprender y entender cómo los métodos de machine learning pueden ser aplicados a sistemas de muchos cuerpos cuánticos tomamos como base el modelo de Bose-Hubbard. Este es un modelo sencillo y bien conocido del que disponemos otros métodos aproximados para resolverlo que se expondrán más adelante. El modelo describe la interacción de bosones que ocupan los sitios de una red n-dimensional. Podemos escribir el Hamiltoniano de este sistema tal que:

$$H = -t \sum_{\langle ij \rangle} (b_i^\dagger b_j + b_i b_j^\dagger) + \frac{U}{2} \sum_i (n_i(n_i - 1)) - \mu \sum_i n_i \tag{2.1}$$

Donde $\langle i, j \rangle$ denota cada par de sitios vecinos en la red, los operadores b_i y b_j son los operadores creación y destrucción de bosones por lo que cumplen que $[b_i, b_j^\dagger] = \delta_{ij}$, y $n_i = b_i^\dagger b_i$ es el operador número. Observamos que el Hamiltoniano se compone de tres términos. El primero es el término de hopping o término cinético, que da cuenta de la movilidad de los bosones en la red, t es la amplitud de salto. El segundo término, U describe el potencial de interacción on-sitio, vemos que sólo es distinto de cero cuando hay más de un bosón y que es repulsivo ($U > 0$). En tercer lugar, tenemos el término del potencial químico, μ.

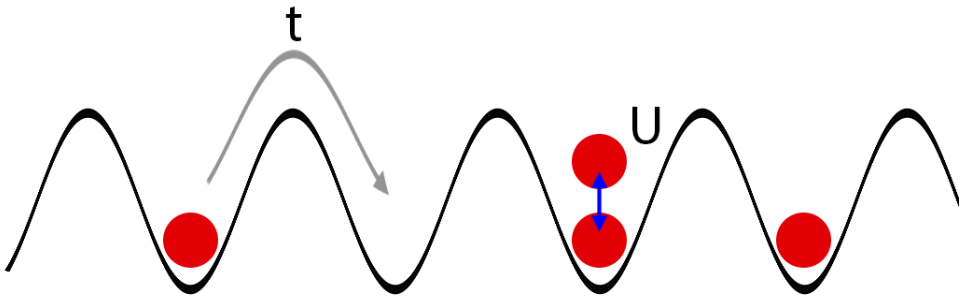

Figura 1: Representación del modelo de Bose-Hubbard en 1D.

Este Hamiltoniano presenta una simetría global $U(1)$, lo cual implica que es invariante bajo una transformación $b_i \to e^{i\theta} b_i$. Esta simetría se rompe espontáneamente para algún valor de U/t. Es decir, podemos observar una transición de fase a temperatura cero entre una fase de aislante de Mott cuando $U/t >> 1$ y una fase superfluida cuando $U/t < 1$.

Para entender las fases que exhibe este modelo vamos a observar los dos casos límite. En la fase de aislante de Mott la interacción entre los bosones en un mismo sitio es grande, y en el límite en el cual $U >> t$ el término de hopping se vuelve despreciable y podemos aproximar el Hamiltoniano como una suma de términos que actúan en un único sitio:

$$H = \frac{U}{2} \sum_{i=1}^{L} (n_i(n_i - 1)) - \mu \sum_{i=1}^{L} n_i = \sum_{i=1}^{L} h_i \tag{2.2}$$

Está claro que

$$h_i \left|n_i\right\rangle = E_i \left|n_i\right\rangle , \quad E_i = \frac{U}{2} n_i(n_i - 1) - \mu n_i \tag{2.3}$$

El estado fundamental será un estado del espacio de Fock, $|n_i\rangle$, con n_i dado por:

$$n_i = max(0, [\frac{\mu}{U}] + 1]) \tag{2.4}$$

De este modo el estado fundamental vendrá caracterizado por un número concreto de bosones por sitio, constituyendo esta fase aislante. De manera intuitiva, si el término de hopping que da cuenta del movimiento de los bosones a lo largo de la red es despreciable, los bosones permanecerán en su sitio y la fase será aislante.

En segundo lugar, si tomamos la situación opuesta en la que los bosones no interactúan entre sí, i.e $U = 0$, el Hamiltoniano se transforma en:

$$H = -t \sum_{\langle ij \rangle} (b_i^\dagger b_j + b_i b_j^\dagger) - \mu \sum_i n_i \tag{2.5}$$

En el caso unidimensional tenemos una cadena lineal, donde si además asumimos condiciones de contorno periódicas, podemos escribir el Hamiltoniano en el espacio de momentos tal que:

$$H = \sum_k (\epsilon_k - \mu) b_k^\dagger b_k, \quad b_k = \frac{1}{\sqrt{N}} \sum_{j=1}^{L} b_j e^- ikj, \quad \epsilon_k = -2tcos(k) \tag{2.6}$$

Donde $k = \frac{2\pi}{N} m$, m $\in \{0, 1, ..., N-1\}$. En este extremo como U=0 no se puede producir una ruptura de la simetría. Y es intuitivo ver que los bosones van a poder moverse libremente por la red sin restricciones ya que no se van a "notar" unos a otros. Esta fase además se caracteriza por una coherencia de fase de largo alcance.

A temperatura distinta de cero se comporta como un fluido regular que no rompe la simetría y no exhibe coherencia de fase. La dimensión del espacio de Hilbert de este modelo crece exponencialmente con el número de sitios en la red, en concreto como $N_{local}^{N_{sitios}}$, por lo que la simulación exacta del espacio de Hilbert está limitada a pequeños sistemas de unas pocas decenas de bosones en unas pocas decenas de sitios, sin embargo los sistemas experimentales que se pueden describir con este modelo tienen un número de sitios varios ordenes de magnitud mayor a las decenas y con una ocupación media superior a la unidad, por lo que es interesante desarrollar métodos numéricos aproximados que nos permitan simular mayores tamaños.

3. Métodos existentes

Con el objetivo de resolver sistemas de muchos cuerpos cuánticos de grandes tamaños se han desarrollado distintos algoritmos y métodos aproximados. En este trabajo en concreto buscamos obtener la energía del estado fundamental del sistema, para lo cual en un sistema sencillo (unos pocos sitios y unos pocos bosones) podríamos simplemente diagonalizar la forma matricial de nuestro Hamiltoniano y obtener sus autovalores, sabiendo que el mínimo de ellos será el que corresponda al estado fundamental.

Sin embargo, el coste computacional de esta tarea crece muy rápidamente conforme aumentamos el tamaño del sistema haciendo inviable obtener el autovalor exacto a partir de unas ciertas dimensiones.

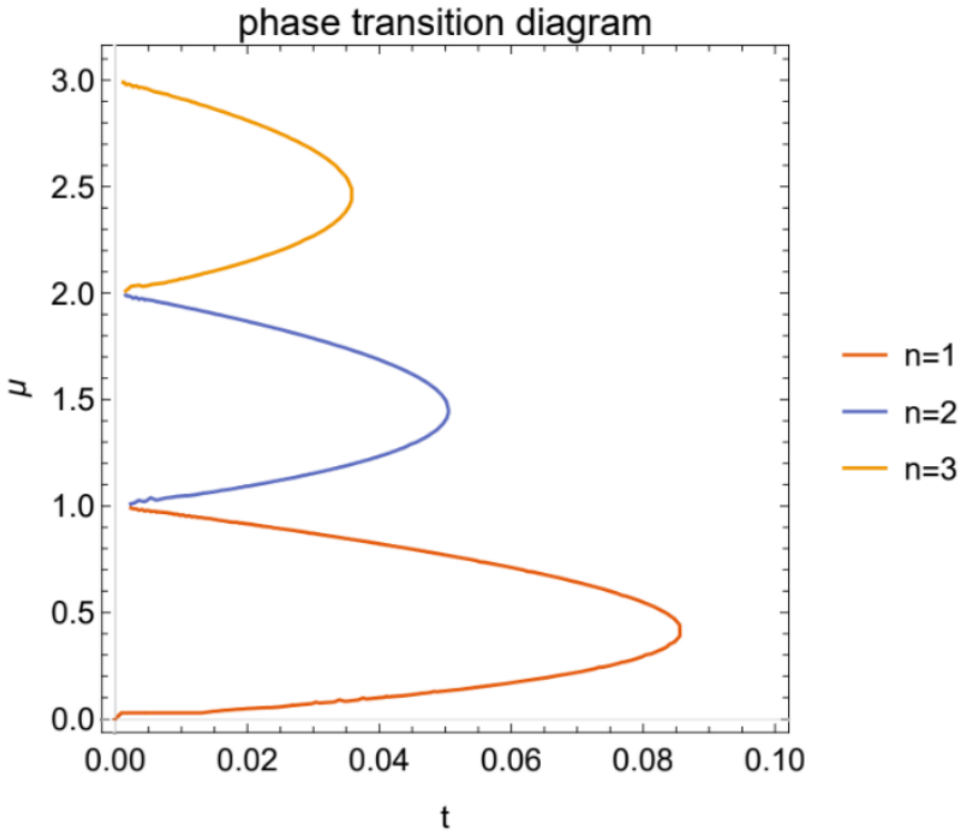

Figura 2: Diagrama de fases del modelo de Bose-Hubbard unidimensional con U=1. Las zonas de los lóbulos pertenecen a la zona de aislante de Mott mientras que las zonas a la derecha de estos son fases superfluidas.

Como en muchos otros tipos de sistemas cuánticos nos enfrentamos a un aumento exponencial de las dimensiones del espacio de Hilbert conforme incrementamos las dimensiones del sistema. En el caso del modelo de Bose-Hubbard, la dimensión del espacio de Hilbert crece exponencialmente como $N_{local}^{N_{sitios}}$ como hemos indicado antes, por lo que, aunque limitemos la dimensión local a unos pocos bosones, diagonalizar de forma exacta la matriz del Hamiltoniano se hace inviable a partir de unas pocas decenas de sitios. Por este motivo resultan de extrema utilidad algoritmos como el de Lanczos o métodos aproximados desarrollados ad hoc para afrontar este tipo de sistemas como el método de Gutzwiller o el uso de ansatz variacionales.

3.1. Algoritmo de Lanczos

El método de Lanczos se ha convertido en uno de los métodos más populares para aproximar unos pocos autovalores de una matriz hermítica. En concreto este algoritmo es el caso particular de la aplicación del algoritmo de Arnoldi para una matriz hermítica. Este tipo de algoritmos entran dentro de los métodos iterativos para calcular valores y vectores propios de matrices, proporcionan generalmente sólo una parte de las soluciones después de un número determinado de iteraciones indicadas por el usuario. Además, estas soluciones son aproximadas y en cada iteración se disminuye el error respecto al valor exacto, de manera que podemos indicar al algoritmo la precisión que deseamos obtener para que este se detenga una vez alcanzada. En general, el objetivo es desarrollar un método que decrezca lo máximo posible el error en cada iteración y con el menor trabajo posible en cada una de ellas.

La idea detrás de este método es proyectar el problema m-dimensional de nuestra matriz del Hamiltoniano en un subespacio de Krylov de dimensión mucho menor. Un subespacio de Krylov es el generado por una matriz A, nxn, y un vector b de dimensión n cuando tomamos como base las imágenes de b sobre las primeras r potencias de A.

Es decir, dada una matriz $A \in \mathbb{R}^{nxn}$ y un vector $b \in \mathbb{R}^{nx1}$, los correspondientes subespacios de Krylov serán los generados por:

$$K_r(A, b) = span\left\{b, Ab, A^2b, A^3b, .., A^{r-1}b\right\} \tag{3.1}$$

Obteniendo así un subespacio de orden r≤n.

El concepto es similar a obtener la descomposición QR de una matriz A, lo cual consiste en obtener la matriz ortogonal Q y la matriz triangular superior R tal que A = QR. Para ello podemos aplicar transformaciones unitarias sobre las filas de A o emplear el método de Gram-Schmidt para producir una base ortonormal del espacio generado por las columnas de A. La idea detrás del algoritmo de Lanczos es análoga a este segundo proceso, tratamos de producir columnas ortonormales que permitan reproducir A en forma Hessenberg (todos los elementos de la matriz son cero por encima o debajo de una de sus subdiagonales).

Una reducción de A a su forma Hessenberg cumple que AQ = QH, pero cuando A es muy grande como en nuestro caso, la reducción completa de A es inabarcable y en lugar de ello se trabaja con m columnas (m ≪ n). De manera que tomamos submatrices de las matrices AQ y HQ, a las que llamaremos Q_m y H_m formadas por las m primeras columnas y las m+1 primeras filas respectivamente:

$$Q_m = [q_1, q_2, ..., q_m], \quad H_m = \begin{bmatrix} h_{11} & h_{12} & \cdots & \cdots & h_{1m} \\ h_{21} & h_{22} & & & \vdots \\ 0 & \ddots & \ddots & & \\ 0 & \cdots & \cdots & \ddots & h_{m+1,m} \end{bmatrix} \tag{3.2}$$

Puesto que AQ = HQ se cumple que $AQ_m = Q_{m+1}H_m$ (porque las filas m+1, m+2, ... de H son cero) y dado que en el caso concreto del algoritmo de Lanczos A es hermítica, H también lo es y por lo tanto tridiagonal:

$$A[q_1, q_2, ..., q_m] = [q_1, q_2, ..., q_m, q_{m+1}] \begin{bmatrix} \alpha_1 & \beta_1 & \cdots & \cdots & 0 \\ h_1 & \alpha_2 & & & \vdots \\ 0 & \ddots & \ddots & & \beta_{m-1} \\ \vdots & \cdots & \cdots & \beta_{m-1} & \alpha_m \\ 0 & \cdots & \cdots & \cdots & \beta_{m+1} \end{bmatrix} \tag{3.3}$$

O equivalentemente

$$Aq_k = \beta_{k-1}q_{k-1} + \alpha_k q_k + \beta_{k+1}q_{k+1}, \quad k = 1, 2, ..., m \tag{3.4}$$

Se puede así obtener a continuación por métodos directos los autovalores de la matriz H_m cuyos autovalores tienen el valor aproximado a la matriz A. Los vectores de Lanczos, q_i se pueden emplear también para obtener los vectores propios aproximados de A. Este proceso se puede implementar en código tal que:

```
v1 = 0
v0 = 0
B1 = 0

for j < (m-1):
    wj = A·vj
    aj = wj·vj
    wj = wj -aj·vj - Bj·vj-1
    Bj+1 = |wj|
    vj+1 = wj/Bj+1

wm = A·vm
am = wm·vm
# Los a's y B's son los coeficientes de la matriz tridiagonal
# Los vectores vj son los vectores de Lanczos
```

Figura 3: Esquema general de un código que implementa las iteraciones de Lanczos.

En definitiva, el método de Lanczos garantiza el cálculo rápido de ciertos valores propios de una matriz hermítica si dichos valores están bien separados del resto del espectro. Los resultados obtenidos con este algoritmo aplicado sobre el Hamiltoniano de Bose-Hubbard han sido más que satisfactorios ya que se reduce ampliamente el tiempo de obtención de la energía fundamental conforme aumentamos el tamaño de la red y el valor numérico no se desvía del obtenido por diagonalización exacta a menos que nos fijemos en la decimotercera cifra decimal ya que trabajamos con una precisión de 10^{14}. Por este motivo decidimos tomar los resultados de Lanczos como una buena aproximación de la energía exacta a la hora de comparar la energía fundamental obtenida mediante el método central de este trabajo.

3.2. Métodos aproximados

Aunque no ha sido objetivo de este trabajo el desarrollar este tipo de técnicas sí que se han tenido en cuenta a la hora de realizar posibles comparaciones y comprender el abanico de posibilidades a la hora de resolver sistemas como el que tratamos de estudiar. Dentro de estos métodos aproximados tenemos principalmente dos. Uno es el método de Gutzwiller, que es el que emplea H.Saito en [3] dónde también resuelve el modelo de Bose-Hubard con redes neuronales. El otro es el uso de ansatz variacionales gaussianos sobre la función de onda. Este segundo método ha sido explorado con más detalle por miembros del departamento de física de la materia condensada de la Universidad de Zaragoza, entre ellos el director de este trabajo.

4. Machine learning y física

Vamos a introducir una serie de conceptos que constituyen el núcleo del trabajo presente. En los últimos años, y con la generalización de las técnicas de aprendizaje automático y big data en la industria, también se ha investigado la posibilidad de emplear redes neuronales para describir estados cuánticos de sistemas de muchos cuerpos debido a la capacidad de estas técnicas de manejar grandes volúmenes de datos. Como ya se ha expuesto anteriormente en el caso de sistemas many-body el principal problema es el conocido como 'maldición de la dimensionalidad' que afrontamos al ir aumentando el tamaño de nuestro sistema y que nos obliga a tratar con espacios de Hilbert de tamaños inabarcables. Con la finalidad de encontrar alternativas a los métodos numéricos y aproximados ya existentes expuestos en la sección anterior se va a presentar ahora otra nueva vía basada en redes neuronales para resolver el modelo de Bose-Hubbard y que es extrapolable a otros modelos many-body.

Por otra parte, para los físicos uno de los objetivos más perseguido dentro de este campo es la capacidad de describir correctamente el estado fundamental del sistema, principalmente porque este es el estado que las partículas van a tender a ocupar inicialmente pero también porque los estados excitados heredarán parte de esta estructura. Esto también nos permite comparar las diferencias entre estados fundamentales en las diferentes fases del modelo.

4.1. Redes neuronales

Una red neuronal artificial (ANN) es un conjunto de neuronas artificiales conectadas entre sí, modelando someramente las conexiones entre neuronas que vemos en un cerebro biológico. Normalmente organizamos estas neuronas artificiales en capas, de manera que cada capa está conectada con la siguiente mediante unos pesos que determinan la intensidad de dicha conexión. La parte fundamental de esta estructura proviene de la no-linealidad presente en las funciones de activación de estas neuronas.

Al final lo que constituye la salida de una neurona artificial es una función no lineal de la suma pesada de todas aquellas neuronas conectadas a esta. Para verlo de una manera sencilla podemos observar la figura 4.

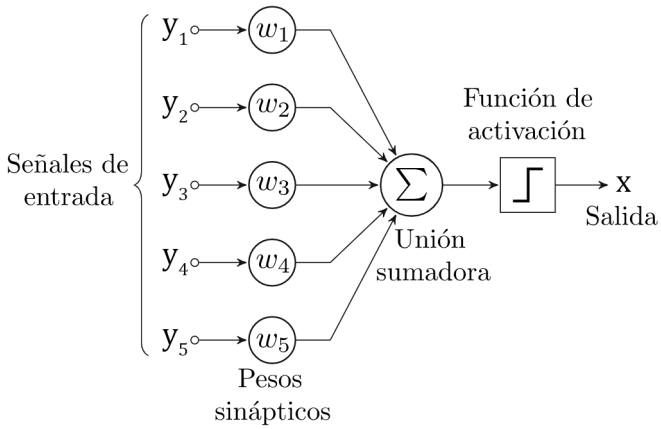

Figura 4: Representación esquemática general de los componentes de una neurona artificial.

Esta función de activación es la que introduce la no-linealidad en el sistema y es la que hace de las redes neuronales algo interesante ya que de otra manera toda esta estructura no sería nada más que una función lineal que relaciona una entrada y una salida, y esto no nos permitiría representar nada interesante. Esta función no lineal puede ser cualquiera, aunque normalmente se eligen funciones sencillas y bien conocidas como la sigmoide, la tangente hiperbólica o reLU (rectified linear unit).

Vamos a ver un pequeño ejemplo de cómo funcionan. Supongamos que tenemos dos capas, teniendo la capa de entrada dos neuronas y una la de salida:

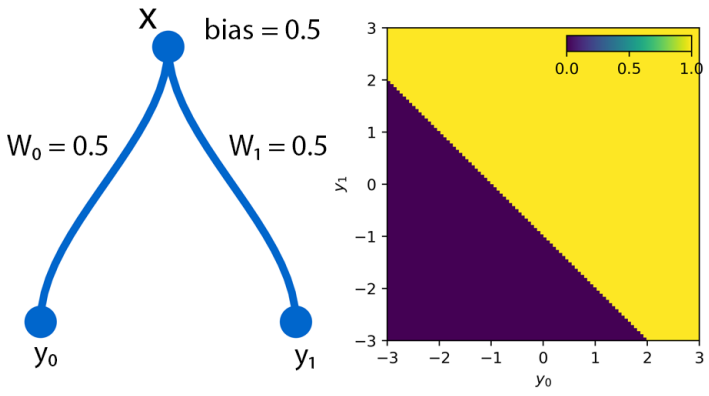

Figura 5: Esquema de una red de tres neuronas y representación de su espacio de salida.

En este caso sencillo trabajamos en un espacio bidimensional que podemos representar fácilmente. Podemos advertir que cada neurona de entrada, que toman los valores y_0 e y_1 respectivamente, está conectada con un peso de la misma magnitud a la neurona de salida. Puesto que estos pesos son iguales la zona de transición en la salida es una recta de pendiente unidad. Por otra parte, la neurona de salida tiene un bias que lo que hace es desplazar la zona de transición. Esta transición es así de inmediata porque hemos usado como función de activación de la neurona de salida una función escalón. Si en vez de usar una función escalón usamos una sigmoide, por ejemplo, la transición sería un gradiente suave. Vamos a ilustrar esto con un segundo, y último ejemplo:

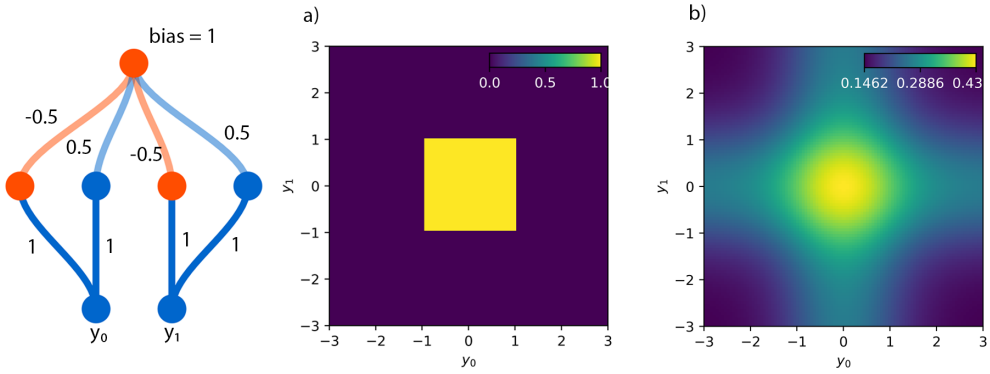

Figura 6: A la izquierda podemos ver la configuración de neuronas que se ha empleado para obtener el espacio de salida a la izquierda empleando dos funciones de activación distintas. En a) se trata de una función escalón y en b) de una sigmoide.

En este segundo ejemplo hemos usado una capa oculta que conecta nuestro espacio bidimensional de entrada y nos permite aprender funciones más complejas. En este caso vemos que la neurona de salida sólo se activa para valores en torno al cero, el área de activación dependerá de los bias de la capa oculta que en este caso son todos de magnitud 1. De manera intuitiva podemos decir que cada una de las cuatro neuronas de la capa oculta aprende uno de los lados del cuadrado que observamos en la salida.

La capacidad de emplear este tipo de herramientas para aproximar funciones de onda y encontrar el estado fundamental proviene de dos puntos. El primero es la capacidad de las redes neuronales de aproximar funciones arbitrarias. Como ya hemos dicho, una red neuronal no deja de ser una función matemática que incorpora un número muy grande de parámetros (pesos y bias de cada neurona) inicialmente no fijados, cuyos valores se buscan mediante métodos de optimización (el aprendizaje por entrenamiento) para obtener una salida deseada frente a una entrada específica. El segundo punto proviene de la física per se y es que no necesitamos más que unos pocos estados, dentro de todas las posibilidades que nos ofrece el espacio de Hilbert, para describir correctamente el estado fundamental.

4.2. Restricted Boltzmann machine

Como se puede intuir, hay muchas maneras de organizar estas neuronas artificiales y sus conexiones en función del problema que deseamos afrontar. Para este trabajo la arquitectura que hemos empleado principalmente ha sido la de una restricted Boltzmann machine (RBM), un tipo de red neuronal estocástica que se ha usado en un amplio abanico de aplicaciones de inteligencia artificial desde su invención en 1986 y la cual es conocida por su capacidad de aprender distribuciones de probabilidad presentes en los datos de entrada.

Las máquinas de Boltzmann se diseñaron como una manera de representar gráficamente una distribución de probabilidad p(v), definiendo unas variables estocásticas (los nodos de la red) v y una energía E mediante la cual estas interaccionan. Las redes de Hopfield fueron las primeras en adoptar este tipo de paradigma y tratar de modelar la memoria humana. El trabajo de Hopfield estableció entonces una relación entre el concepto de aprendizaje y el de optimizar la energía de una red neuronal (los parámetros que la definen). Mientras que las redes de Hopfield tienen un número de nodos igual al tamaño del sistema físico que se trata de estudiar, las máquinas de Boltzmann extienden este espacio para poder incluir nodos e interacciones más allá de los físicos. Este espacio oculto o latente, lo llamaremos h (de hidden en inglés), es capaz de capturar correlaciones de mayor orden entre los nodos visibles de la entrada, mejorando la capacidad de la red neuronal de representar la distribución p(v,h). Para facilitar el aprendizaje de esta estructura se restringió la interacción entre los nodos de un mismo tipo, es decir, sólo hay interacción entre nodos visibles y ocultos y nunca entre ellos mismos dentro de una misma capa. Así obtenemos la estructura de una RBM, conocida por ser un aproximador universal. Es decir, dado un número suficiente de nodos en la capa oculta, la red es capaz de representar aproximadamente cualquier distribución de probabilidad a una precisión arbitraria.

En concreto el ansatz que se ha empleado en este trabajo para reproducir la función de onda del modelo de Bose-Hubbard es:

$$\Psi(n_1, n_2, ..., n_L) = e^{\sum_i^L a_i n_i} \times \prod_{j=1}^{M} \cosh\left(\sum_i^N W_{ij} n_i + b_j\right) \tag{4.1}$$

Donde a_i, b_j y W son los valores de los bias visibles, los bias ocultos y la matriz de pesos que conecta ambas capas de la RBM respectivamente. Podemos hacernos una idea más intuitiva de la estructura de la RBM empleada para esta tarea en la figura 7.

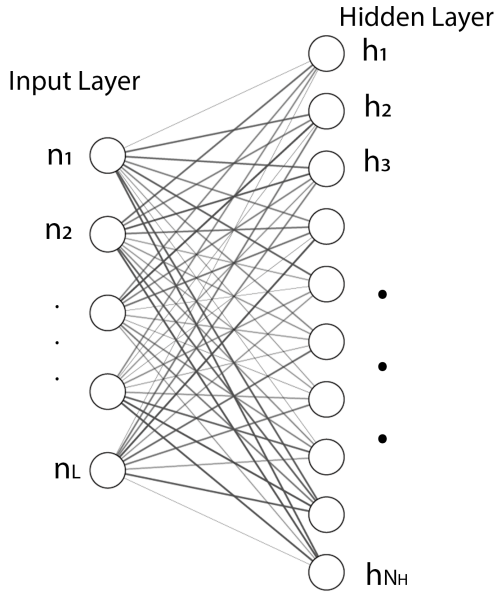

Figura 7: Podemos apreciar la sencilla estructura de una RBM. Esta se constituye de una capa de unidades visibles n_i y una capa de unidades ocultas h_i. Los parámetros adicionales que definen la energía de la máquina son sus pesos y sus bias, estos los podemos entender como la intensidad de la conexión entre las neuronas y los vemos representados aquí según su opacidad.

4.3. Variational Monte Carlo

Una vez hemos explicado la posibilidad de emplear una RBM como un ansatz para representar estados cuánticos lo natural es preguntarse cómo conseguimos hacer que esta aprenda el estado fundamental de nuestro sistema. Buscamos la aproximación del ansatz en función de los coeficientes de la RRNN a la f.d.o, lo cual se consigue ajustando este set de parámetros $\{a_i, b_j, W_{ij}\}$ de manera que se minimice el valor esperado de la energía $\langle\psi| H |\psi\rangle$.

Como se puede intuir, estamos haciendo uso del principio variacional, que nos dice que dado un sistema cuyo Hamiltoniano es H, si ψ es cualquier función normalizada que cumple las condiciones de contorno del problema, entonces:

$$|\Psi\rangle = \sum \Psi(n_1, n_2, ...) |n_1, n_2, ...\rangle = \Psi(\mathbf{n}) |\mathbf{n}\rangle \tag{4.2}$$

Donde n_i es el número de bosones en cada i-esimo sitio. De este modo cuando le damos un set de enteros $\mathbf{n}$ a la RBM obtenemos el valor de la aproximación de la f.d.o para esa configuración $\mathbf{n}$. Hay que tener en cuenta que para realizar esta simulación en un ordenador debemos imponer un límite a este número de bosones en cada sitio, a este número le llamaremos N_L, ya que es el tamaño de la dimensión local del espacio de Hilbert en un único sitio, y será la dimensión de los operadores creación y destrucción en cada sitio. Con esta base construimos un proceso de Monte Carlo variacional con un método de sampling que puede variar en función del problema a afrontar.

En este trabajo hemos empleado tanto un método de Metropolis como un método exacto, la diferencia entre ambos es que con Metropolis a cada iteración en el algoritmo de optimización se samplea una configuración $\mathbf{M}$ con probabilidad $\frac{|\psi(n)|^2}{\sum_{n'}|\psi(n')|^2}$ mientras que si empleamos un sampler exacto lo que hacemos es explorar todas las configuraciones posibles en el espacio de Hilbert. Este segundo método es obviamente mucho más costoso computacionalmente y sería inabarcable en sistemas de gran tamaño, pero nos puede servir para detectar problemas o aplicar mejoras en el resto del código. A partir de este proceso de sampling podemos obtener el valor esperado de la energía calculado estocásticamente como:

$$\left\langle \sum_{n'} \langle \mathbf{n} | \, H \, | \mathbf{n'} \rangle \, \frac{\Psi(\mathbf{n'})}{\Psi(\mathbf{n})} \right\rangle_M = \langle H \rangle_M \tag{4.3}$$

Donde $\langle ... \rangle_M$ implica el promedio sobre el Metropolis sampling de $\mathbf{n}$. De este modo podemos optimizar los parámetros $\mathbf{W}$ y $\mathbf{h}$ de la red para conseguir que el valor esperado de $\langle H \rangle$ sea mínimo, lo más cercano posible al estado fundamental. Para conseguir esto podemos emplear cualquiera de los numerosos optimizadores disponibles dentro del paquete NetKet como AdaGrad o Rms. Esto implica calcular la derivada de la energía respecto a los parámetros de la red, lo cual resulta en

$$\frac{\partial \langle H \rangle}{\partial \omega} = 2Re \left[\frac{\sum_{n,n'} \frac{\psi^*(\mathbf{n})}{\partial \omega} \langle \mathbf{n} | \, H \, | \mathbf{n'} \rangle \, \psi(\mathbf{n'})}{\sum_n |\psi(\mathbf{n})|^2} - \langle H \rangle \frac{\sum_n \frac{\psi^*(\mathbf{n})}{\partial \omega} \psi(\mathbf{n})}{\sum_n |\psi(\mathbf{n})|^2} \right] \tag{4.4}$$

Donde ω denota un parámetro cualquiera de la red. De este modo los parámetros se pueden actualizar durante el entrenamiento tal que

$$\omega \rightarrow \omega - \gamma \frac{\partial \langle H \rangle_M}{\partial \omega} \tag{4.5}$$

donde γ es el ratio de actualización de los parámetros (learning rate). El valor de γ se toma del orden de 10^{-1} a 10^{-3}, el promedio para obtener $\langle ... \rangle_M$ en cada actualización se toma de un total del orden de 10^3 samples. Los parámetros de la RBM se inicializan de forma aleatoria con una desviación estándar de 0.1.

Para generar este sample $\mathbf{M}$ se propone un estado inicial aleatorio n0, y se continúa proponiendo un nuevo estado n'. La probabilidad de aceptar dicho cambio $n \rightarrow n'$ será $|\Psi(\mathbf{n'})|^2/|\Psi(\mathbf{n})|^2$. Así, para construir $\mathbf{M}$ se realiza un número de actualizaciones indicado por el usuario. Podemos ilustrar este proceso de Monte Carlo empleando Metrópolis sampling de la siguiente manera:

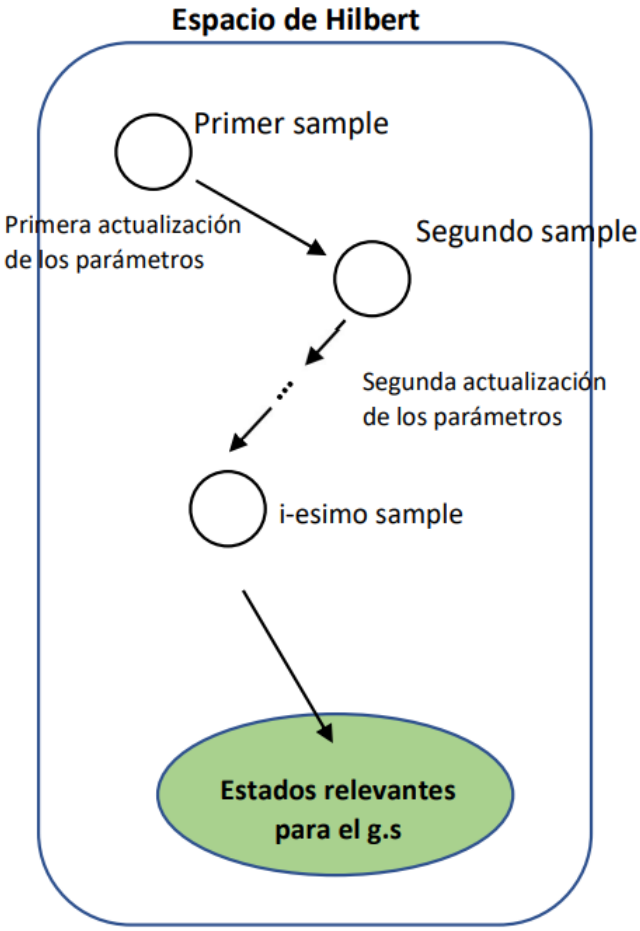

Figura 8: Representación intuitiva del proceso de Metrópolis

4.4. NetKet

Finalmente debemos indicar que para implementar las distintas partes que hemos ido explicando hasta este punto hemos empleado una librería open source de Python llamada NetKet, desarrollada por Carleo et al. [6] con el objetivo de ofrecer un framework sencillo de utilizar a la hora de estudiar sistemas de quantum many-body con técnicas de machine learning.

5. Resultados

Durante el desarrollo de este trabajo hemos realizado pruebas con distintos ansatz (RBM, FFNN), métodos de sampling (Metropolis, Exact) y de los hiperparámetros de los mismos (número de steps, densidad de la capa oculta...) y a continuación exponemos los resultados más relevantes que hemos obtenido y su comparación con otros de los métodos existentes.

5.1. Densidad de la capa oculta

Hemos comentado previamente que una RBM es un aproximador universal, y que podemos aproximar una función con una precisión arbitraria siempre y cuando aportemos un número suficiente de unidades ocultas. Esto es algo que me gustaría presentar a continuación. Se ha usado un tamaño de lattice de 5 sitios con dimensión local igual a 4, estableciendo los parámetros del Hamiltoniano de Bose-Hubbard en la fase de aislante de Mott, con $U/t = 0{,}4$. Empleando el sampler exacto podemos analizar la diferencia entre tener una densidad de unidades ocultas mayor o menor:

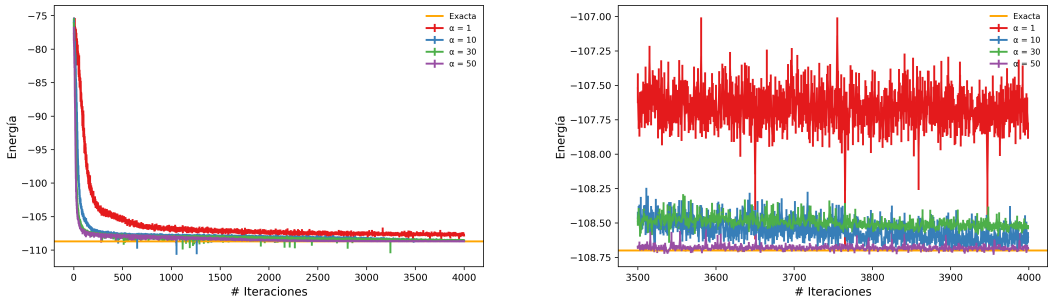

Figura 9: Comparativa en la energía durante el entrenamiento de una RBM con distintas densidades de la capa oculta. A la derecha vemos la ampliación en las últimas 500 iteraciones.Parámetros: $U = 2, t = 5, \mu = 14$.

Vemos que aumentar la densidad de las unidades ocultas mejora el ritmo al que converge a la energía fundamental así como mejorar la precisión del estado final en el que termina. Sin embargo esta mejora notable en los resultados implica un mayor tiempo de entrenamiento ya que hay más parámetros que optimizar en la RBM. Por este motivo lo interesante es encontrar un punto donde los resultados sean lo suficientemente buenos sin emplear tiempos de entrenamiento innecesariamente largos.

5.2. Energía

A continuación buscamos comparar la energía fundamental alcanzada tanto por Lanczos como por la RBM. Los datos a continuación han sido obtenidos con una lattice de 5 sitios con una dimensión local de 4. Podemos observar como en todos los regímenes la RBM se comporta bien y aproxima correctamente el valor esperado de la energía fundamental. Se ha empleado una densidad de la capa oculta de 40 por lo que la precisión llega hasta la primera o segunda cifra decimal, aunque hay ciertos valores de U/t para los cuales esta precisión es mucho mayor, como podemos observar en la figura:

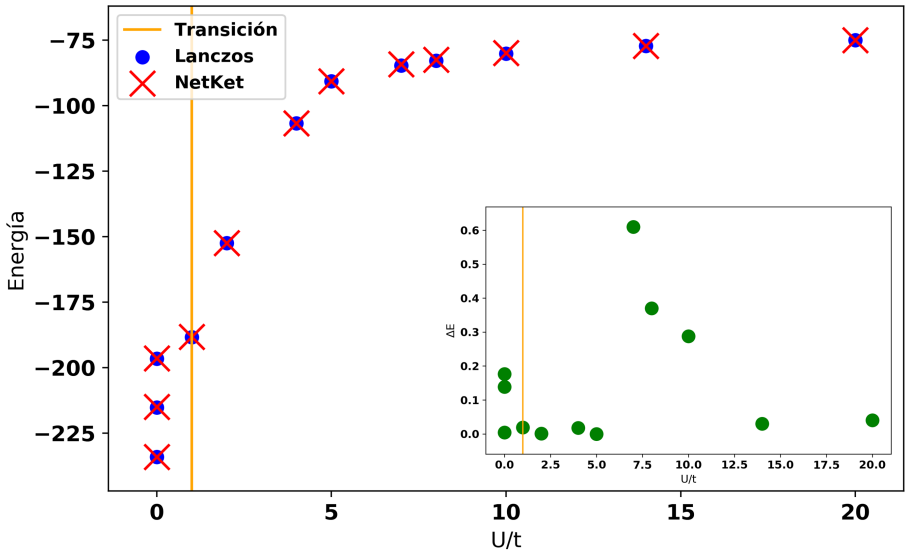

Figura 10: Comparativa de la energía mínima obtenida mediante una RBM y mediante Lanczos. Los resultados presentan una gran precisión en unos casos mientras que en otros la diferencia es mayor. Un comportamiento parecido se puede apreciar en los resultados de Saito en [3].

5.3. Rendimiento

Puesto que una de las problemáticas que presenta resolver de manera exacta sistemas con un espacio de Hilbert tan grande es el propio tamaño de la matriz, es interesante ver cómo de eficiente es cada método en el uso de memoria. Para ello se han monitorizado el uso de memoria en una instancia n1-standard-16 de Google Cloud Platform donde disponemos de 16 núcleos y 64 GB de RAM:

Nsitios	Nlocal	Dim matriz	Exacta [GB]	Lanczos [GB]	NetKet [GB]
4	3	81	0,003	0,0067	0,012
6	4	4096	0,38	0,019	0,036
7	4	16384	6,06	0,033	0,048
8	4	65536	Crash	0,089	0,043
9	4	262144	Crash	0,194	0,069
10	4	1048576	Crash	0,650	0,093
10	5	9765625	Crash	8,160	0,234
12	4	16777216	Crash	15.810	0,774
13	4	67108864	Crash	Crash	1,660

Cuadro 1: Comparación de uso de memoria en GB.

Lo primero que debemos comentar es que la diagonalización exacta de la matriz se ha realizado con el método 'exact' que ofrece la librería NumPy. Esta al parecer no procesa las matrices en forma sparse y esto provoca que sea incapaz de alojar toda la matriz en memoria a partir de una cierta dimensión, dándonos un error. Esto es algo que podíamos esperar antes de hacer la prueba.

Es destacable la comparación entre Lanczos y NetKet. Si bien ambos necesitan de la construcción de la matriz previamente y se produce un pico de uso de memoria en ese momento, durante la aproximación del estado fundamental la RBM de NetKet nos ofrece una mejora sustancial en el uso de memoria, algo muy a tener en cuenta a la hora de estudiar sistemas muy grandes. La diferencia se aprecia muy bien si representamos la tabla como en la figura 11.

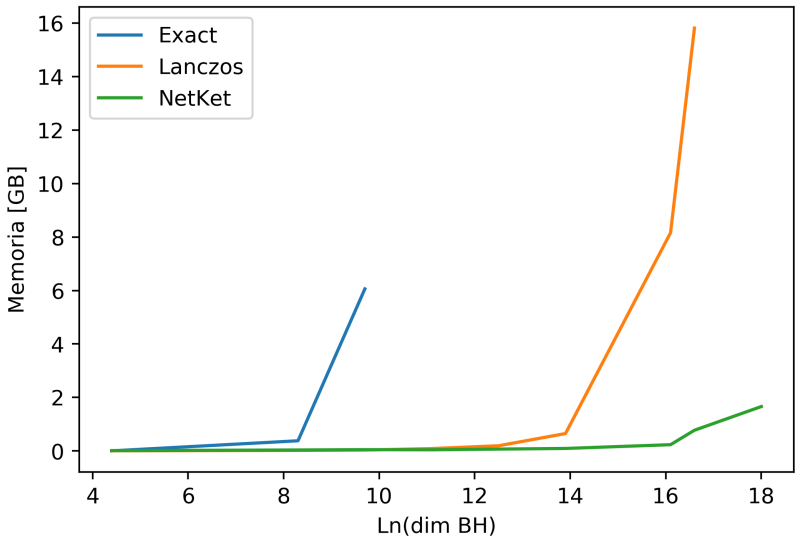

Figura 11: Comparación en el uso de memoria entre los tres métodos. Observar que el eje horizontal está en escala logarítmica.

Observamos que el método exacto que implementa Numpy basado en las rutinas LAPACK, que aplican un algoritmo QR sin manejar la matriz en forma sparse, se queda sin memoria rapidamente. Por otra parte tenemos el algoritmo de Lanczos que nos ofrece NetKet haciendo uso de la librería SciPy, el cual sí maneja las matrices de forma sparse y vemos que nos ofrece buenos resultados hasta que alcanzamos el sistema con 13 sitios y dimensión local 4, en el cual este método también se queda sin memoria, a pesar de estar usando un servidor con 64GB, lo cual a fecha de hoy está muy por encima de las capacidades de memoria de los ordenadores domésticos promedio.

Finalmente debemos comentar que NetKet se comporta excepcionalmente bien en términos de uso de memoria, ya que en definitiva no tiene que almacenar grandes matrices sino los parámetros de la RBM, haciendo de este método aproximado una opción mucho más asequible a la hora de estudiar sistemas de gran tamaño. El inconveniente que hemos encontrado es la compatibilidad con el entrenamiento en paralelo de la RBM, lo cual sí funciona en mi portátil personal pero por problemas de compatibildad ha fallado en el servidor, haciendo que los tiempos de entrenamiento fuesen inusualmente largos. Pero estos son inconvenientes técnicos más sencillos de solucionar.

6. Conclusiones

Tenemos claro que el sampler exacto va a ser el que mejor funcione, pero no tiene ningún sentido tratar de emplearlo siempre, por lo que lo ideal sería desarrollar un sampler ad hoc para el problema. Carleo ya explica que el sampler de MetropolisLocal no es el mejor si tenemos un potencial químico y sugiere emplear otro como el MetropolisHop. Sin embargo este último no está implementado en NetKet.

Otra pregunta que debemos plantear es si NetKet es una opción óptima para realizar una investigación y avances en el campo. Creo que es una opción válida para iniciarse y probablemente funcione muy bien en los sistemas para los que ha sido bien diseñada y testeada. Sin embargo, la documentación no está completa ni del todo clara. Además, probablemente programando todo uno mismo adaptado al problema de interés se puedan obtener mejores resultados, tal y como hizo Saito[3], sin embargo fue el propio Saito quien nos recomendó el uso de NetKet.

Finalmente, hemos demostrado que las RRNN sirven para este cometido, pero ¿reemplazaran métodos existentes? Mi opinión es que para encontrar el estado fundamental pueden ser muy útiles si se consigue un método de entrenamiento y de sampling eficiente y capaz de ejecutarse en paralelo, luego es muy probable que con el desarrollo de estas herramientas y del deep learning veamos cada vez más estos métodos aproximados a la hora de afrontar sistemas de muchos cuerpos cuánticos. Si bien es cierto que hay un sacrificio en la precisión de la energía obtenida en nuestros experimentos, es sin duda interesante probar también nuevos ansatz que puedan ofrecer mejores resultados. además de todo esto, las ANN parecen ser una herramienta más profunda y se pueden emplear para entender correlaciones dentro del sistema y propiedades más allá del estado fundamental.

Referencias

[1] Jorge De La Calleja y Olac Fuentes. «Machine learning and image analysis for morphological galaxy classification». En: *Monthly Notices of the Royal Astronomical Society* 349.1 (2004), págs. 87-93.

[2] Giuseppe Carleo y Matthias Troyer. «Solving the quantum many-body problem with artificial neural networks». En: *Science* 355.6325 (feb. de 2017), págs. 602-606. ISSN: 1095-9203. DOI: 10.1126/science.aag2302. URL: http://dx.doi.org/10.1126/science.aag2302.

[3] Hiroki Saito. «Solving the Bose-Hubbard Model with Machine Learning». En: *Journal of the Physical Society of Japan* 86.9, 093001 (sep. de 2017), pág. 093001. DOI: 10.7566/JPSJ.86.093001. arXiv: 1707.09723 [cond-mat.dis-nn].

[4] Dan Guest, Kyle Cranmer y Daniel Whiteson. «Deep learning and its application to LHC physics». En: *Annual Review of Nuclear and Particle Science* 68 (2018), págs. 161-181.

[5] Muhammad Imran Razzak, Saeeda Naz y Ahmad Zaib. «Deep learning for medical image processing: Overview, challenges and the future». En: *Classification in BioApps*. Springer, 2018, págs. 323-350.

[6] Giuseppe Carleo y col. «NetKet: A Machine Learning Toolkit for Many-Body Quantum Systems». En: *SoftwareX* (2019), pág. 100311. DOI: 10.1016/j.softx.2019.100311. URL: http://www.sciencedirect.com/science/article/pii/S2352711019300974.

[7] F Arute, K Arya y R Babbush. «Quantum supremacy using a programmable superconducting processor». En: *Nature 574, 505–510 (2019)* ().